Security for Distributed SOA at the Tactical Edge

R. William Maule
Naval Postgraduate School
rwmaule@nps.navy.mil

William C. Lewis
OPNAV N2/N6FC12
bill.c.lewis@navy.mil

ABSTRACT

Service-Oriented Architecture (SOA) presents new paradigms for security and information assurance. Security processes for legacy systems are not sufficient to secure a distributed SOA. Yet, many of the new services arriving at the tactical edge for Fleet testing are based on web services—meaning that SOA is somewhat inevitable. This paper addresses some of the challenges encountered when testing SOA in live Fleet operations and in controlled laboratory experiments. Discussion of SOA capabilities is followed by a case study that uses them to determine the source of a latency failure under expected user tactical load. The case study concerns performance issues encountered in both field and laboratory experiments when implementing certificates to secure web services in distributed service-based tactical workflow automation.

INTRODUCTION

Comprehensive assessment of Service-Oriented Architecture (SOA) components, applications and services requires operational enterprise-class architecture against which to test proposed configurations. Models and simulations are not adequate for SOA security evaluation. Measured variables from actual SOA operations on live networks need to be assessed and documented. SOA Evaluation Architecture (SEA) was developed to help assess security in distributed SOA, including delineation of SOA variables and their associated security risk metrics. As a test facility for SOA, SEA has implemented both enterprise-class and best-of-breed SOA solutions, in multiple configurations.

Service-Oriented Architecture (SOA) presents an opportunity to significantly increase overall information technology effectiveness and performance, while substantially reducing hardware, software, and programming costs. However, as with any new technology, there will be unforeseen costs and risks. Security is one such category of risk and the impacts can be considerable—from external threats, and from internal processes—an example of which is documented in this report. When fully developed, SEA will provide hard data for assessments of new services, quality of those services, and their associated security risks.

SOA SERVICES

As an example of utility, consider SOA software applications and composite applications. SOA software applications are defined as services that can be called from other applications on other servers, using machine-to-machine (M2M) communications [1]. In theory, a service that has been once defined can be connected to service consumers with very little effort. This situation will lead to an overall increase in network use, which will in turn lead to increasingly complex interactions [2]. Security analysis will need to address services, components, objects, and data types [3, 4]. Service level security contracts will need to formalize component relationships, object interactions, and associated rules [5].

So, while SOA dramatically reduces information production costs through shared resources [6], and there is a significant increased efficiency from shifting information management responsibilities closer to those responsible for accuracy of the data, the cumulative savings are somewhat offset by increased costs to upgrade the network and security systems to support SOA.

Services can build upon one another to form composite applications. For example, several different services—each providing information on a different track or target—could be assembled into a composite service that displayed all tracks and associate data from the different sources. Since the data is sent in XML format, the composite service can additionally parse the cumulative data, add additional logic, and provide the unified resource as a single service—which can in turn be published for others to use.

Composite services should provide composite service users with a level of service integration that had previously been cost-prohibitive. The SOA composite application approach, however, introduces complexity and security risk, as a failure at any step of the process could render the results incorrect, or vulnerable. Such failures have been a common occurrence in recent operational experiments.

To describe the intended assessment of SOA through SEA, this paper examines contemporary widely used models and components for SOA implementation. This includes variables and options within each SOA model and component, and discusses considerations for operational, technical, and security assessment.

For a Fleet implementation, several operational assumptions have been used during this analysis:
1. Current Navy SOA implementations are disjointed, resulting in cost over-runs; absence of comprehensive functioning architectures has resulted in “stovepipe” SOAs and also resulted in integration requirements that arguably negate expected cost benefits and efficiencies.
2. Significant cost reductions for information technology hardware, software and personnel can be achieved using SOA and supporting technologies (such as virtualization and cloud computing with its shared storage, memory, and CPUs). However, such reductions cannot be achieved given current implementation patterns.

3. Technical capabilities of the Navy could increase greatly by SOA automation of currently manual information processes, and by SOA mediated customization of content and user interfaces. The SOA Software as a Service (SaaS) model could reduce costs for software modifications and customizations across the entire software life-cycle—if effectively implemented. SOA support by cloud computing environments could reduce costs, increase information technology effectiveness, and significantly reduce security risks.

4. SOA design can support both comprehensive backbone services and forward deployed networks with specialized, flexible configurations for platforms and units. Effective design requires recognition that different SOA models are required to accommodate different applications.

Additionally, functional requirements are assumed in this analysis:

A. Security will be a foremost concern since compromise of a SOA and its machine-to-machine processes would compromise every service and server across that SOA. Virtualization technologies increase technical efficiencies but also introduce a new and unknown level of security risk. Cloud computing offers the potential to significantly reduce maintenance and security costs but would also introduce the possibility of a simultaneous cloud-wide security breach.

B. Operations will necessitate distributed SOA and nodes that remain fully operational even during communication outages—with SOA able to synchronize content without operator involvement when communications are restored. Multiple levels of SOA will be required to meet the needs of the Navy; a “one-size fits all” approach to SOA would not be in the best interests of the Navy. Different operational units may require different types of SOA to meet their operational needs and communication contexts.

SEA TEST ARCHITECTURE

This paper describes an instance of the above architecture type as implemented in hardware in the SEA environment. SEA uses actual DON link measurements and commercial SOA to generate realistic SOA behaviors. Major facets of SEA have been tested in fleet operations.

SOA is applied by DOD as an integral component of the Global Information Grid (GIG), by DISA as Net-Centric Enterprise Services (NCES), by Navy through FORCEnet and CANES, by the Air Force as C2 Constellation, and by the Army within Future Combat Systems (FCS). The SEA test environment can evaluate DON applications, both deployed and backbone (“cloud”).

In Figure 1, the number “1” nodes are the “cloud” data centers on the backbone, the number “3” nodes the ship and forward-deployed units with limited bandwidth and communications, and the number “2” nodes the at-sea configurations that provide intermediary services. Number “2” nodes are typically located in a network operations center (NOC), or on a command ship or carrier.

![Figure 1. SEA notional infrastructure.](image)

Specifically, a backbone SOA node, number “1”, is a comprehensive SOA that has been optimized for broadband communications and high availability. Backbone SOA nodes would serve major commands, Maritime Domain Awareness (MDA) and Maritime Headquarters (MHQ), and provide the Navy interface to the Global Information Grid (GIG). Shore-based sites would include Europe and other locations with a major Network Operating Center (NOC) and robust, high availability communications.

Second tier service communications are not as robust as a backbone SOA, but try to provide continuous online communications with shore nodes. Examples would include the reliable movement of information over unreliable links. Rules would be established to carry only real-time, strategic, tactical, and operational information on the second tier, while high-level policy, reference and archival information remain ashore, on tier 1 [assuming that rules and policies are necessary to optimize communications but a complicating variable when scaling SOA].

Third tier communications, identified with the number “3”, are both low bandwidth and unreliable. SOA optimized for track- and target- prioritized communications over low bandwidth links would be used by smaller platforms in a battle group, by submarines, and by shore groups with second- or third-tier communications (such as USMC tactical units). Initial SEA tests simulated all three tiers of Figure 1 above, and integrated infrastructure services (such as directory and security services).

SOA for each afloat or deployed unit will assure continued information processing when communications are down, and will also provide an ability to synchronize both content and infrastructure services after communications are restored. Content synchronization among distributed SOA
nodes adds considerable complexity and risk to scaling, and was simulated in the tests below to assess risk because of its tactical usefulness.

In Figure 2, circled letters are implementation types, with the backbone nodes “A” and “B” to provide the comprehensive suite of SOA components and services. These implementations would be in “cloud” data centers, access the comprehensive directories and security systems, and maintain the master metadata repository. Letter “C” indicates a system with second tier communications capabilities, likely the carrier in a battle group or other platform that serves as the battle group network center. Letter “D” would be bandwidth-challenged environments.

Note that many of the DOD and Navy systems do not use a “full stack” of SOA infrastructure services. Some have a custom set of SOA infrastructure, specific to the applications. Others have only web services. Some technology sponsors offer a customized or “reduced stack” SOA that lacks traditional core SOA capabilities (e.g., to optimize throughput in bandwidth-limited environments). For example, sensor data for Intelligence, Surveillance, and Reconnaissance (ISR) may flow over a limited bandwidth link; Common Operational Picture (COP) applications may be legacy; MDA biometric data may be transposed to XML, and then processed through a partial SOA stack.

In the tested architecture, security and directory services are managed centrally from a “primary node”, with provisions for backup and live failover. In more detail, there are three (3) general levels of coordination required for SOA-to-SOA synchronization. The first level contains security and directory services that identify users and authorize their information access. The second level coordinates application flows through and across SOAs, generally through the Enterprise Service Bus or a similar technology. The third and last level contains the content synchronization achieved at the storage, database, or application layers. A glimpse into this complexity, as implemented in the SEA test environment, is presented in Figure 3 – which expands Figure 2 to show how ship deployed and battle command SOA were integrated in SEA. Security is implemented across SOA nodes, between SOA nodes, and at every service integration point.

In Figure 3, clouds labeled with the number “1” are single node, stand-alone SOA, for example a backbone data center. Domain controllers can route information to each cloud’s physical location, process “A” in the diagram. Security services are integrated to provide users and machines with single sign-on (SSO) across clouds – process “B”. Ongoing efforts in the DOD address comprehensive SOA security but to date there is no DOD standard. Several options were explored in the SEA project. Process “C” denotes end-to-end security for web services. Each of these processes, “A”-“C”, is needed for communications in a secure SOA DOD environment.

Security and user directory integration are one of the most labor consuming facets of systems integration, and will be an issue when scaling SOA. Specific techniques are required to integrate and synchronize user privileges for web services, applications, and content when integrating SOA nodes. These specific techniques are neither well established nor proven, especially for the hybrid SOA models common in the DOD. An ESB serves as the application router for SOA and techniques to integrate the ESBs of both similar and dissimilar SOA (illustrated as processes “D”-“F”) will present a challenge (as addressed in OASIS reference document SOA-RM [7]). Security setting propagation between applications through the ESB is similarly complex.

The node or cloud designated with the number “3” in Figure 3 presents a mobile “reduced” SOA, without the full interoperability or integration stack. In addition to the Intelligence, Surveillance and Reconnaissance (ISR), Common Operational Picture (COP), and Maritime Domain Awareness (MDA) examples given above – which require either hybrid or custom SOA – a similar “slimmed down” approach is sometimes used commercially. Analogs from the commercial sector would include online transaction processing that uses real-time servers to process thousands of simultaneous transactions per second. Examples of such online transaction processing include the Automated Teller Machine (ATM) inputs for a bank, and Radio Frequency Identification (RFID) scan inputs for a large retailer. The military analog would be processing fused data into sensor tracks for a command and control system.

Data analysis is at the other extreme from real time processing and high bandwidth data streams. Data analysis involves a few users, each running resource-intensive services. An example would be real-time data stream mining, perhaps for analysis of transactions. The military analog would be intelligence analysis that synthesizes large
sets of real-time data to estimate future intentions [e.g., of a terrorist group].

The command ship “2” in Figure 3 represents the integration point (at sea) for battle group operations, and the backbone SOA node “D” represents the integration point (on land) for operations across all Fleets. Node or cloud “2” depicts a configuration appropriate for a command ship. In configuration this node is smaller than a backbone node, but is larger and has more components than the forward-deployed, reduced stack SOA. Node “2” is optimized for communication, able to act as the integration point for a battle group, and is the interface with the shore backbone “D”. This sea-based SOA, given best-available communications to shore, would collect data from forward-deployed units and ships in a battle group (such as “3”), and send data back to them. We simulate communications within the battle group that are independent of satellites.

**SEA TEST ENVIRONMENT**

Now that available features have been presented, consider a case study of their application.

**A. Case Study**

In this case study, a fairly realistic SOA system is loaded until latencies become excessive, at which point the above available features are used to determine what SOA component caused the excessive latency failure.

1) **General approach**

In building a SOA, one useful assessment technique is to take a baseline then re-test as components are added to see if any latency has changed. If the new latencies are accepted, then those new latency measures become new baselines. When new web services are added they are referenced against the new baseline, and so on. In this manner it is possible to determine, at an early stage, when something has gone awry with the SOA at the infrastructure level. We will apply this general technique to a security component, and then assess the security implications.

The initial test architecture is presented in Figure 4 and includes an ESB, BPM, Rules Engine, and Security framework – choreographed through BPEL. The diagram is substantially accurate but omits two items: the UDDI Registry and the BPM suite. Neither is essential to understanding the test.

The baseline test configuration was complex enough to be realistic. It featured managers and control centers for
both Grid and virtualization. It also contained fixed and mobile hosts and clients, wireless and broadband hard-wired lines, and both firewalled and LAN connections, as shown in Figure 4.

2) Equipment and software, to include load generation.

Oracle Fusion Middleware with BPEL, ESB, and a Rules engine were mounted on a BEA WebLogic application server, which was in turn mounted to an Oracle database. Sun Java Server was used for the tests. Governance was handled through an Oracle web services Policy Manager. The test scenario employed a web service tier and messaging process flow through a workflow automation application. Coordination was through BPEL, a Rules Engine, ESB, and Business Activity Monitor (BAM). These components comprised the load-set for the experiment. Users and processes were then scaled – at different bandwidths, with different loads on those bandwidths. All processes were operating within manufacturer specifications.

The test loaded the Figure 4 network, and compared loaded latencies against the unloaded initial baseline ranges to assess loaded performance. Loads included user submission of data through a web service data form interface, automated and manual workflows for approvals, and external processing through the BPEL, Rules Engine, and ESB. Processes were monitored through the BPEL console, ESB, and BAM. Simulated user data was filed to a directory and, through an adapter, processed into the ESB – where it was fed back into a BPEL system and BAM (Figure 5):

1. User data submitted through a form interface
2. BPEL data, processed under pre-defined rules
3. Real time data, monitored by a BPEL in real-time
4. User input data, automatically archived as XML
5. Files, routed by the ESB to BPEL and BAM

This workflow was selected for testing because it offered direct parallels to several of the tactical planning applications currently being tested in Sea Trial experiments. The general workflow is as follows: After data input, automated processes make decisions such as firing on a target, or making an acquisition. These decisions are subject to manual approval before implementing the decision. This specific workflow was also selected because it was designed by the software’s vendors, so there can be some assurance of proper coding practices and quality control.

3) Procedure

After completion of the workflow setup procedure, and several successful runs of the workflow [which included both automated and human decision points], the system was loaded and placed into a stress test. The stress test was to assess overall security and performance latencies under medium load. In addition, the WS-Security open standard was enabled, with security information in the header of all SOAP messages.

The initial runs of the stress test imposed a light workload and proceeded without incident. Ten (10) processes were enacted concurrently, with each process regenerating five (5) times (Figure 6) to approximate user and load scaling over time. A latency of 1000ms was established between the users and the servers through network impairment software to approximate medium-latency satellite communications. Stress test configuration settings were left at the software vendor’s default. The tests were without incident, and as shown in Figure 6, completed within an average of 784ms, with minimum of 117ms and maximum of 1797ms.

Next, workload was tripled over the previous test. The system was loaded with three (3) cascading batch jobs (in lieu of the 1 previous), with each job consisting of 10 processes, each regenerating 5 times. The SOA slowed by an order of magnitude, as documented in “Instance 2 stress test aggregate reports” (Figure 6). The average process time increased more than tenfold (10x) to render the system nearly unusable. Minimum latency was similar, but maximum latency was about twenty (20x) larger for a very significant difference in overall processing rate. System logs revealed a token (security) error during verification, likely from latency in the certification process under load. Significant latency in retrieval from the data store was evident at service invocation.
Discussion

1. A fully operational SOA, with a manufacturer-supplied workflow process, operating within manufacturer specifications, was base-lined then given a stress test.
2. Latencies were measured on “best-of-suite” manufacturer-integrated, loosely-coupled components.
3. Manufacturer integration was expected to be efficient. Third party systems integration was not required so can be ruled out as a causal factor in stress induced failure.
4. Under light load, a moderately complex workflow operated normally with no errors or rogue processes.
5. Stress tests were enacted. The initial stress test loaded the SOA workflow by simulating intensive interaction by 10 users, each user conducting multiple tasks. The SOA operated without incident.
6. A second stress test simulated 30 simultaneous users, each invoking multiple web services. Under this load, excessive latency was measured in the data store responsible for holding and parsing the security token.
7. Examination of the logs revealed an error somewhere in the security process related to a security token.

Some further explanation is warranted. As previously discussed, security is the most complex facet of a SOA and the area most likely to generate problems. This has been shown repeatedly in SOA experiments in Fleet operations. Also to know is that when a process fails, or a small sub-process in this case, that failure may not be obvious to users. Failure does not simply kill the workflow or application. Rather, a Java routine known as “garbage collection” automatically kills the process, but can then attempt to recover from the error by restarting the process. Users may not even be aware that a process has failed, only that a particular function has been slower in a given instance. Often network latency is blamed for delays when in reality a Java process is attempting to compensate for an error. To optimize SOA operations, reasons for these failures should be resolved, so that they occur much less often.

CONCLUSION

Investigation of SOA problems generally requires forensic analysis. Such analysis for SOA or web service operations is extremely rare in the DON and its execution requires personnel with networking, systems administration, and SOA practical experience. Forensics is also very time-consuming, especially when the problem is intermittent. In this example, an educated guess would be that the cumulative stress tests compounded and overloaded the security system. Basically, the CPUs could process faster than the token could be passed or authenticated. Considerable additional testing would be needed to validate this hypothesis. At a minimum, security issues are very significant for any SOA in which all data must be encrypted and secured, and for SOAs which must support large numbers of simultaneous users.

REFERENCES